Remarks

Reconsideration of this Application is respectfully requested.

Upon entry of the foregoing amendment, claims 1-73 are pending in the application, with 1, 27, 40, 53 and 67 - 71 being the independent claims.

These amendments are believed to introduce no new matter, and their entry is respectfully requested.

In the Office Action dated June 4, 2007, the abstract is objected to. Claim 59 is objected to. Claims 67-68 stand rejected under 35 U.S.C. § 101 due to allegedly being directed to non-statutory subject matter. Claims 1-7, 9, 10, 13-29, 31, 32, 35-45, 48-55, 57, 58, 67-73 stand rejected under 35 U.S.C. § 102(b) as being allegedly anticipated by Duesterwald et al., U.S. Patent Publication No. 2003/0101330. Claims 8, 11, 12, 30, 33, 34, 46, 47, 56, 59 and 60 stand rejected under 35 U.S.C. § 103(a) as being allegedly unpatentable over Duesterwald et al.

Based on the above amendment and the following remarks, Applicants respectfully request that the Examiner reconsider all outstanding objections and rejections and that they be withdrawn.

Objection to the Abstract

The abstract has been amended, to reduce the number of words to fewer than 150, as required by the Examiner. No new matter has been added. Applicants respectfully request that the objection to the abstract be withdrawn.

Objection to Claim 59

Claim 59 is objected to due to an incorrect dependency. The dependency of this claim has been corrected. Applicants respectfully request that the objection be withdrawn.

Rejections under 35 U.S.C. § 101

Claim 67 and 68 stand rejected under 35 U.S.C. § 101. To address this rejection, Applicants are amending these claims to recite a computer usable recording medium. Applicants respectfully submit that this amendment overcomes any possible 35 U.S.C. § 101 issues, and respectfully request that the § 101 rejections be withdrawn.

Rejections under 35 U.S.C. § 102(b)

All of the independent claims stand rejected under 35 U.S.C. § 102(b) as being allegedly anticipated by Deusterwald. These rejections are respectfully traversed. The Office Action relies on Deusterwald as allegedly disclosing "mark instructions" and the use of the mark instructions as a counter. Specifically, the Office Action relies on paragraph 48 of Deusterwald as allegedly disclosing this aspect. Respectfully, this is incorrect. Deusterwald does not disclose either of these aspects. Paragraph 48 of Deusterwald is reproduced below:

> [0048] **As the various code fragments are executed by the application 102 under the control of the DELI 100, the DELI "sees" each piece of code that is executed. Through the monitoring process, the DELI 100 can, therefore, determine which code fragments are used most frequently.** The DELI 100 can then make the determination of which pieces of code are "hot," i.e., most important to application execution with reference to the policies that are provided by the system control and configuration layer 112. As noted above, this determination can be made using program counters that track execution instances. Persons having ordinary skill in the art will appreciate that various other methods can be used to make the determination of which pieces of code are hot. **Examples of the manner in which this determination can be**

> **made are described in U.S. patent application Ser. No. 09/186,945, filed Nov. 5, 1998, entitled "Method for Selecting Active Code Traces for Translation in a Caching Dynamic Translator," and U.S. patent application Ser. No. 09/312,296, filed May 14, 1999, entitled "Low Overhead Speculative Selection of Hot Traces in a Caching Dynamic Translator," both of which are hereby incorporated by reference into the present disclosur**e.

Deusterwald does not deal with individual instructions as such – Deusterwald deals in "fragments," which are groups of instructions, or functions (procedures). Deusterwald uses the terms "fragments" and "traces" synonymously:

> [0026] Irrespective of the manner in which control is obtained over the application 102, an instruction fetch controller 128 can then be used to extract (i.e., fetch) **copies of fragments (e.g., traces)** of the application binary code, pass them to the DELI core 106 for caching, and direct the core 106 to execute the appropriate cached copies out of its code cache(s) 124. Use of the transparent mode layer 110 in facilitating such operation is described below in relation to FIG. 4.

Also, Deusterwald, in paragraph 48, refers to U.S. Patent Application No. 09/186,945, which has issued as U.S. Patent No. 6,351,844. "Traces" are also defined in the '844 patent as follows:

> A trace is a **sequence** of dynamic instructions characterized by a start address and a branch history which allows the trace to be dynamically disassembled.

Thus, a trace, or fragment, which is the basic unit with which Deusterwald deals, is a sequence of instructions, or, using the definition in Deusterwald, a function or a procedure:

> [0053] As described above, there are several problems with current methods of dealing with unavailable, for example faulty or missing, hardware functionality. These problems can be avoided, however, when the DELI 100 is used in that the DELI controls very small portions of code such as code fragments and even individual instructions. In operation, the DELI 100 can be used to copy code fragments from an application 102 and determine which call upon faulty or missing hardware functionality. When such code fragments are "detected," the DELI 100 can dynamically replace them with new code

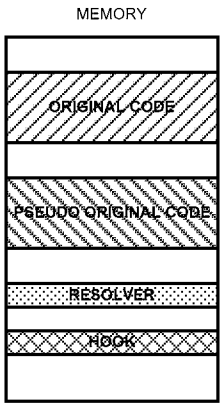
> fragments that do not require that functionality. The new code fragments can be cached such that, **next time the original code fragments (i.e., a particular function)** are required, the new code fragment(s) can be executed within the code cache(s) 124 to bypass the faulty or missing functionality. Notably, where many code fragments are copied to the code cache(s) 124, substantially all execution may ultimately occur within the code cache(s).

This is self evidently not the same thing as a single, standalone instruction. More importantly, Duesterwald makes no distinction between instructions that can be used as countable "mark" instructions and any other instruction – thus, even if a function contains only one instruction (which would be a degenerate case), Duesterwald does not require that single instruction to be a countable instruction – it can be any instruction. It is worth noting that the absolute majority of instructions in a processor's instruction set cannot be used as mark instructions. In sum, Duesterwald lacks the concept of mark instructions as that term is defined in the specification and recited in the claims.

Furthermore, the approach used in Deusterwald is fundamentally different from what is claimed. Referring again to the '844 patent, which is incorporated by reference in the Deusterwald specification, the '844 patent is directed to binary translation – this is a technique known in the context of virtualization and virtual machines, which are also discussed at length in the '844 patent. In essence, the traces, or fragments, are executed entirely under control of another entity, such as a virtual machine monitor. In this case, execution of the instructions can always be interrupted by a program with a higher privilege level (i.e., the virtual machine monitor), and in a manner that is completely transparent to the code that is being executed. This,

however, is an entirely different approach to the problem than counting the number of mark instructions and patching (or not patching) the code, as claimed in the independent claims.

Additionally, Duesterwald does not have pseudooriginal code. At a basic level, there are four components at play: original code, pseudooriginal code, a resolver (see, e.g., claim 9), and a hook. These components are loaded into memory:



Duesterwald has original code, a resolver (of sorts) and a hook – i.e., code that does "something different" from the original code (without the hook, the whole purpose of patching is pointless). But Duesterwald does NOT copy original instructions to another location in memory to generate pseudooriginal instructions. Duesterwald only has the hook – but not pseudooriginal instructions.

Applicants believe that this distinction alone is sufficient to overcome Duesterdwald, and respectfully request reconsideration. Applicants also note that claims 1 and 71 are not amended in any way, to avoid the next Office Action being a final office action (particularly in view of the new rules coming into effect on November 1, 2007 regarding continuing applications). Applicants traverse the rejection of claims 1 and 71 solely on the merits.

Additionally, in order to advance the prosecution of this case, Applicants have amended all the independent claims except claims 1 and 71 to recite that the mark instructions are countable "**wherein a number of times the mark instructions have been executed is countable in order to determine how many of the original instructions should be executed if control returns from an external execution context to the original instructions addresses after patching.**" This addresses the situation where a particular block of original code is relatively long, and so many of the original instructions have been executed, that at this point, patching this block of original code can lead to unpredictable results. This aspect is not disclosed anywhere in Deusterwald, and therefore provides an additional, separate, ground for patentability for the independent claims.

Thus, for all of these reasons, Applicants respectfully submit that Deusterwald does not disclose mark instructions, nor does Duesterwald disclose pseudooriginal code, nor does Deusterwald use any instructions for determining whether or not the original instructions can be patched. In essence, for all of these reasons, Duesterwald does NOT guarantee that patching will always be done correctly. Reconsideration is therefore requested for these reasons.

Rejections of claims 9, 31 and 57

Regarding dependent claims 9, 31 and 57, the Office Action relies on paragraph 48 as allegedly disclosing the aspect of "the modified instructions including a resolver to determine a number of the instructions at a location of the original code that had already been executed." The Office Action relies on paragraph 48, specifically, on the phrase "DELI100 sees each piece of code that is executed. Through the monitor in process, the DELI100 can, therefore, determine

which code fragments are used more frequently." Respectfully, this passage (and, in fact, the entire paragraph 48, as well as the rest of Deusterwald) does not disclose the claim limitations.

Deusterwald counts how often **code fragments** – i.e., groups of instructions – are executed. Deusterwald does not count how often individual instructions are executed – and Duesterwald does not use the instructions themselves for counting. Also, Deusterwald does not modify the replacement code to do this. Reconsideration is therefore respectfully requested based on this additional ground as well.

Applicants also note that claim 9 is not amended in any way, also to avoid the next Office Action being a final office action. Applicants traverse the rejection of claim 9 solely on the merits.

Rejections under 35 U.S.C. § 103(a)

All of the claims rejected under 35 U.S.C. § 103(a) over Deusterwald are believed to be allowable at least because their independent claims are allowable.

Rejections of claims 3 and 5

Additionally, Applicants respectfully disagree with the Office Action's reasoning regarding dependent claims 3 and 5, see page 7 of the Office Action. The Office Action argued:

> Duesterwald discloses:
>
> > - prior to the allocating step, masking interrupts (see at least paragraph 45 "DELI **100** is injected into the application **102** with the injector **126** ... gains control over the application and its execution" - This can consider masking the interrupt of the application).

Respectfully, the Office Action is incorrect. Duesterwald is not masking interrupts – rather Duesterwald is replacing certain system calls in the code with other system calls that are

compatible with the hardware for which Duesterwald is patching the code. Reconsideration is respectfully requested based on this additional, separate ground.

Applicants also note that claims 3 and5 are not amended in any way, also to avoid the next Office Action being a final office action. Applicants traverse these rejections solely on the merits.

Rejections of claims 11, 33, 46 and 59

Additionally, Applicants respectfully disagree with the Office Action's reasoning regarding dependent claims 11, 33, 46 and 59, see page 18 of the Office Action. The Office Action argued:

> However, it would have been obvious to one having ordinary skill in the art at the time of the invention was made to recognize that after the instructions had already been executed, the contents of the instructions had already been changed. So, if there is an interrupt occurs during the execution process, the hook is going to execute before the unchanged instructions of the original code not the changed instructions. Otherwise the patching result would be different than expected.
>
> Therefore, one would have been motivated to add this step in the patching process to ensure the patching results are correctly.

First, the handling of interrupts is unrelated to the question of patching – the problem that is being solved in the present application is not patching of instructions while an interrupt occurs (that is a totally separate problem), but rather patching of instructions that are in the process of being executed. Thus, interrupts have nothing to do with the claim limitations, and Applicants respectfully submit that the analysis in the Office Action is flawed.

Aside from that, Applicants respectfully disagree with the obviousness analysis, particularly where the obviousness arguments rely on the same Duesterwald reference as the §

102 rejections. The claims at issue recite that "if the number of instructions that had already been executed is less than a number of original instructions to be patched, the resolver calls the copied instructions at the storage location so as to imitate a "no patch installed scenario."" This is not at all an obvious way to address this problem.

For example, one relatively "painless" alternative could be to simply ignore this issue, on the theory that an event such as this might happen rarely, if ever. Another possibility is to keep patching the code regardless of how many instructions have been executed. Yet another possibility is to patch the code later, after it has been executed, and the current address in the instruction pointer register has moved to a different memory address. Yet another possibility would be to place the code inside a virtual machine, trigger an interrupt, and patch the code while the code is "running under control" of a virtual machine monitor. Yet another possibility is to detect the problem, and wait until the code is no longer in memory, but rather patch the code only on the disk (and warn the user of this event). In sum, there are numerous possibilities for dealing with this situation, and what the Office Action proposes is not at all an obvious choice. Reconsideration is therefore respectfully requested for claims 11, 33, 46 and 59 based on this ground as well.

Applicants also note that claim 11 is not amended in any way, also to avoid the next Office Action being a final office action. Applicants traverse the rejection of claim 11 solely on the merits.

Reconsideration and allowance of this application is respectfully requested.

Conclusion

All of the stated grounds of objection and rejection have been properly traversed, accommodated, or rendered moot. Applicants therefore respectfully request that the Examiner reconsider all presently outstanding objections and rejections and that they be withdrawn. Applicants believe that a full and complete reply has been made to the outstanding Office Action and, as such, the present application is in condition for allowance. If the Examiner believes, for any reason, that personal communication will expedite prosecution of this application, the Examiner is invited to telephone the undersigned at the number provided.

Prompt and favorable consideration of this Amendment and Reply is respectfully requested.

Respectfully submitted,

BARDMESSER LAW GROUP

/GB/

George S. Bardmesser
Attorney for Applicants
Registration No. 44,020

Date: September 3, 2007

910 17th Street, N.W.
Suite 800
Washington, D.C. 20006
(202) 293-1191